In: Proceedings of the Eleventh Annual International Symposium on Algorithms And Computation (ISAAC 2000), Lecture Notes in Computer Science 1969, pp. 228–240.

Simple Algorithms for a Weighted Interval Selection Problem

Thomas Erlebach[1] and Frits C.R. Spieksma[2]

[1] Computer Engineering and Networks Laboratory, ETH Zürich, CH-8092 Zürich, Switzerland, erlebach@tik.ee.ethz.ch

[2] Department of Mathematics, Maastricht University, P.O. Box 616, NL-6200 MD Maastricht, The Netherlands, spieksma@math.unimaas.nl

Abstract. Given a set of jobs, each consisting of a number of weighted intervals on the real line, and a number m of machines, we study the problem of selecting a maximum weight subset of the intervals such that at most one interval is selected from each job and, for any point p on the real line, at most m intervals containing p are selected. This problem has applications in molecular biology, caching, PCB assembly, and scheduling. We give a parameterized algorithm GREEDY$_\alpha$ and show that there are values of the parameter α so that GREEDY$_\alpha$ produces a $\frac{1}{2}$-approximation in the case of unit weights, a $\frac{1}{8}$-approximation in the case of arbitrary weights, and a $(3-2\sqrt{2})$-approximation in the case where the weights of all intervals corresponding to the same job are equal. Algorithm GREEDY$_\alpha$ belongs to the class of "myopic" algorithms, which are deterministic algorithms that process the given intervals in order of non-decreasing right endpoints and can either reject or select each interval (rejections are irrevocable). We use competitive analysis to show that GREEDY$_\alpha$ is an optimal myopic algorithm in the case of unit weights and in the case of equal weights per job, and is close to optimal in the case of arbitrary weights.

1 Introduction

We study a *weighted job interval selection problem*, called WJISP. The input consists of jobs, each of which is given by a set of intervals on the real line, and a number m of available machines. (We use "WJISP$_m$" instead of "WJISP" if we want to make an explicit reference to the number m of machines.) Each interval i has a positive weight $w(i)$. A feasible solution is a subset of the given intervals such that (1) at most one interval is selected from each job, and (2) for any point p on the real line, at most m intervals overlapping p are selected. The goal is to find a feasible solution that maximizes the sum of the weights of the selected intervals. We let n denote the total number of intervals in the input. We assume that a sorted list of all interval endpoints is available; such a list can be constructed in time $O(n \log n)$.

The requirement that any point on the real line is overlapped by at most m selected intervals is equivalent to the requirement that the selected intervals can be partitioned into m subsets such that the intervals in each subset are pairwise disjoint. In some applications this partition (in addition to specifying the selected intervals) is required as output. However, the subproblem of computing such a partition given the selected intervals can be solved efficiently by coloring the corresponding interval graph. In fact, n intervals can be colored in time $O(n)$ if the sorted list of interval endpoints is given. We concentrate on the problem of selecting the intervals and assume that, if required, an appropriate coloring procedure is employed to compute the partitioning.

WJISP$_m$ has applications in diverse areas. Problems in printed circuit board assembly (see [7], [12]), in molecular biology ([5]) and in time-constrained scheduling on identical machines ([3, 4]) each give rise to instances that can be modelled as (special cases of) instances of WJISP (see [8] for details).

Motivated by these applications we distinguish a number of variants of WJISP that concern the weights of the intervals:

- the *unweighted case* (called JISP) where each interval has the same weight w,
- *WJISP with equal weights per job* refers to instances of WJISP in which intervals that belong to the same job have the same weight, but intervals that belong to different jobs can have different weights, and finally
- *WJISP with arbitrary weights*.

We consider a class of simple deterministic algorithms for WJISP, and we investigate so-called *worst-case ratios* (or approximation ratios) that can be obtained using algorithms from this class. Using standard terminology (see e.g. [11], [1]) we say that an algorithm for WJISP achieves (approximation) ratio ρ if it always outputs a feasible solution whose weight is at least as large as ρ times the weight of an optimal solution. We exhibit an algorithm from this class that achieves a ratio that equals the best possible ratio for algorithms within this class in the unweighted case and in the case with equal weights per job. In case of arbitrary weights, the algorithm achieves a ratio close to the best possible ratio.

1.1 Known Results

Theoretical work on WJISP has largely been confined to ratios for polynomial time algorithms. The unweighted version of WJISP$_1$, called JISP$_1$, is studied in [12]. It is shown that JISP$_1$ is MAX SNP-hard even if every job contains only two intervals and all intervals have the same length. Furthermore, it is shown that the value of the natural LP relaxation of JISP$_1$ is at most twice the value of an integral optimum, and a simple greedy algorithm that achieves approximation ratio $\frac{1}{2}$ is presented.

Several authors have considered a machine scheduling problem with hard deadlines that is closely related to WJISP. In that application, a job consists of all (infinitely many) time intervals of the required length between its release time and its deadline. Bar-Noy et al. [3] give algorithms for m identical machines with ratio $1 - 1/(1 + 1/m)^m$ in the unweighted case (using a greedy algorithm) and in the case with arbitrary weights (using linear programming), and an algorithm with ratio $1/3$ for unrelated machines. They also present a combinatorial algorithm ADMISSION with running-time $O(mn^2 \log n)$, where n is the number of jobs, that achieves approximation ratio $3 - 2\sqrt{2}$ for jobs with arbitrary weights on unrelated machines. Berman and DasGupta [4] present a combinatorial two-phase algorithm with ratio $1/2$ for unrelated machines and arbitrary weights and with ratio $1 - 1/(1 + 1/m)^m$ for m identical machines, thus matching the LP-based bounds of [3]. Bar-Noy et al. [2] use the so-called *local ratio technique* to obtain a combinatorial two-phase algorithm with approximation ratio $1/2$ for unrelated machines and arbitrary weights as well. The results of $[3, 4, 2]$ also apply to WJISP$_m$.

The on-line variant of this machine scheduling problem is studied in [9] and [10] in the single-machine case. The weight of a job is equal to its length and the algorithms receive the jobs in order of non-decreasing release times. Preemption is not allowed. In [9], a deterministic algorithm with ratio $1/2$ if all jobs have the same length and a randomized algorithm with expected ratio $\Omega(1/\log c)$ if the ratio of the longest to the shortest job length is c are presented. In [10], better bounds are derived for the case that the slack of a job is at least proportional to its length.

1.2 Our Results

A natural class of algorithms to consider for WJISP instances is the class of *single-pass* algorithms. Very generally stated, single-pass algorithms are algorithms in which a feasible solution is obtained by iteratively making a decision concerning an item or an object. The first-fit decreasing algorithm for the bin packing problem and the nearest neighbor algorithm for the traveling salesman problem are prime examples of single-pass algorithms. This kind of algorithms can be useful since they need little computing time and/or little information (i.e., they can be applied in an on-line setting). In our context, we call an algorithm a single-pass algorithm when given some sequence of the intervals, each interval is (iteratively) either rejected or accepted (selected) without considering the intervals that will be processed later. Rejections are permanent, but an accepted interval can be rejected (preempted) at a later time. At any time, the set of currently selected intervals must be a feasible solution. After the last interval is presented, the set of currently selected intervals is taken as the solution computed by the algorithm.

When considering a specific single-pass algorithm for WJISP, it is crucial to specify the mechanism that determines the sequence in which the intervals will be processed. We investigate the special class of deterministic single-pass algorithms that process the intervals in order of non-decreasing right endpoints. We call these algorithms *myopic algorithms*. These algorithms seem to be the simplest algorithms for which constant approximation ratios can be achieved.

Analyzing myopic algorithms for WJISP can be seen as studying an on-line problem. Thus, we can use competitive analysis to prove lower bounds on the best approximation ratio that can be achieved by any myopic algorithm. In applications where the intervals correspond to time periods, an on-line scenario in which the algorithm receives the intervals in order of non-decreasing right endpoints may appear unnatural. For instances where all intervals have the same length, however, the order of the left endpoints and the order of the right endpoints coincide. So the concept of myopic algorithms applies to the "real" on-line problem for such instances. For the machine scheduling problem discussed in Sect. 1.1, "the special case of all jobs having the same length under the arbitrary delay model is of great interest" (quoted from [9]).

Table 1 describes our results. In Sect. 2, we give an algorithm called GREEDY$_\alpha$ that can be implemented to run in $O(n^2 \min(m, \log n))$ time. GREEDY$_\alpha$ achieves the ratios described in Table 1; each of these ratios is tight. In Sect. 3, we prove that no myopic algorithm achieves better ratios than the ones described in Table 1 under "Lower Bound". (Notice that these "lower bounds" do actually mean that any myopic algorithm

Table 1. The results

Variant of WJISP	Ratio of GREEDY$_\alpha$	Lower Bound
JISP	$\frac{1}{2}\ \forall m \geq 1$	$\frac{1}{2}\ \forall m \geq 1$
WJISP with equal weights per job	$3-2\sqrt{2} \approx 0.172\ \forall m \geq 1$	$3-2\sqrt{2} \approx 0.172 \quad m=1$
WJISP with arbitrary weights	$\frac{1}{8}\ \forall m \geq 1$	$\approx \frac{1}{7.103} \approx 0.141 \quad m=1$

achieves an approximation ratio smaller than the given values; nevertheless, we refer to these bounds as "lower bounds", because this term is commonly used for negative results.) We emphasize that our lower bounds for myopic algorithms apply only to deterministic algorithms, not to randomized algorithms.

2 Algorithm GREEDY$_\alpha$

We propose a myopic algorithm called GREEDY$_\alpha$, shown in Fig. 1, as an approximation algorithm for WJISP. It has a parameter α that can take (meaningful) values in the range $[0, 1]$. GREEDY$_\alpha$ considers the intervals in order of non-decreasing right endpoints. It maintains a set S of currently selected intervals. When it processes an interval i, it computes a set $C_i \subseteq S$ such that i could be selected after preempting the intervals in C_i and such that C_i has minimum weight among all such sets. C_i is called the *cheapest conflict set* for i. The algorithm selects i only if $w(C_i) \leq \alpha w(i)$, i.e., if the total weight of selected intervals increases by at least $(1-\alpha)w(i)$ if i is selected and the intervals from C_i are preempted. The algorithm can be seen as a parameterized variant of the general double-the-gain paradigm. We can prove the following result:

Theorem 1. *For JISP$_m$, GREEDY$_\alpha$ achieves approximation ratio $\frac{1}{2}$ for any α in the range $[0, 1)$. For WJISP$_m$ with equal weights per job, GREEDY$_\alpha$ achieves approximation ratio $\frac{\alpha(1-\alpha)}{1+\alpha}$. For WJISP$_m$ with arbitrary weights, GREEDY$_\alpha$ achieves approximation ratio $\frac{\alpha(1-\alpha)}{2}$. These results are tight.*

Proof. Let A be the set of intervals returned by GREEDY$_\alpha$, and let T be the set of intervals that were selected by GREEDY$_\alpha$ at least at some time during the execution. If all intervals have the same weight, GREEDY$_\alpha$ never preempts a selected interval (provided that $0 \leq \alpha < 1$), so we have $A = T$ in case of JISP$_m$. In general, we have

$$w(A) \geq (1-\alpha)w(T), \tag{1}$$

because GREEDY$_\alpha$ selects a new interval i only if the total weight of selected intervals increases by at least $(1-\alpha)w(i)$.

Let OPT be an optimal solution. We charge the weight of the intervals in OPT to intervals in T and then prove a bound on the total weight charged to every single interval in T. In this way, we can bound $w(OPT)$ in terms of $w(T)$ and, using (1), in terms of $w(A)$. The details of the charging depend on the variant of WJISP.

Algorithm GREEDY$_\alpha$

$S = \emptyset$; { set of currently accepted intervals }
for all intervals, in order of non-decreasing right endpoints **do**
  $i =$ current interval;
  $C_i =$ minimum-weight subset of S such that $(S \setminus C_i) \cup \{i\}$ is feasible;
  if $w(C_i) \leq \alpha w(i)$ **then**
    $S = (S \setminus C_i) \cup \{i\}$;
  fi;
od;
return S;

Fig. 1. Algorithm GREEDY$_\alpha$

Consider an interval $i \in OPT$. If $i \in T$, we charge $w(i)$ to i. If $i \in OPT \setminus T$, consider the time when GREEDY$_\alpha$ processed interval i. Let C_i denote the minimum-weight set of intervals whose removal from S would have allowed to accept i. Let C'_i be the subset of C_i that contains all intervals that do not belong to the same job as i. If S contains an interval from the same job as i, denote that interval by i' and note that $C'_i = C_i \setminus \{i'\}$; otherwise, we let i' be an imaginary interval with zero weight (just to simplify the formulas) and observe that $C'_i = C_i$.

Let Q_i denote the set of all intervals in S that intersect i and that do not belong to the same job as i. As S is feasible, Q_i can be partitioned into m sets $Q_{i1}, \ldots, Q_{im}$ of intervals such that the intervals in each set $Q_{i\ell}$ are pairwise disjoint. Note that

$$w(i') + w(Q_{i\ell}) > \alpha w(i) \text{ for } 1 \leq \ell \leq m, \qquad (2)$$

because of the definition of C_i and because GREEDY$_\alpha$ did not accept i.

First, consider the case of JISP$_m$. All intervals have the same weight. We charge $w(i)$ to the interval in S that belongs to the same job, if such an interval exists; otherwise, we charge $\frac{1}{m}w(i)$ to an arbitrary interval from each of the sets $Q_{i\ell}$ (note that all the sets $Q_{i\ell}$ must be non-empty in this case). As every interval j receives charge at most $w(j)$ from intervals of the same job and charge at most $m \cdot \frac{1}{m}w(j)$ from intervals overlapping the right endpoint of j, we have that $w(OPT) \leq 2w(T) = 2w(A)$.

Now consider WJISP$_m$ with arbitrary weights. We charge $\min\{w(i), \frac{1}{\alpha}w(i')\}$ to i'. If the remaining weight $w(i) - \min\{w(i), \frac{1}{\alpha}w(i')\}$ is positive, we divide it into m equal parts and distribute each part among the intervals in one set $Q_{i\ell}$, where the charge one interval in $Q_{i\ell}$ receives is proportional to the weight of that interval. More precisely, interval $j \in Q_{i\ell}$ is charged

$$\frac{w(i) - \min\{w(i), \frac{1}{\alpha}w(i')\}}{m} \cdot \frac{w(j)}{w(Q_{i\ell})} .$$

We can deduce from (2) that every interval $j \in Q_i$ is charged by i for at most $\frac{1}{\alpha m}w(j)$ in this way. Altogether, every interval $j \in T$ can receive charge at most $\frac{2}{\alpha}w(j)$ (charge at most $\frac{1}{\alpha}w(j)$ from an interval belonging to the same job and charge at most $\frac{1}{\alpha}w(j)$ from intervals in OPT that overlap the right endpoint of j), implying that $w(OPT) \leq \frac{2}{\alpha}w(T) \leq \frac{2}{\alpha(1-\alpha)}w(A)$.

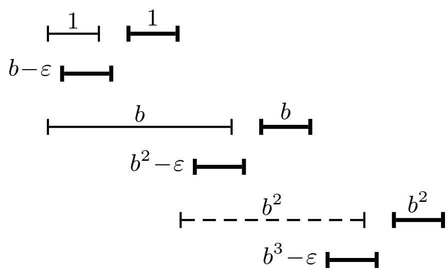

Fig. 2. Worst-case example for GREEDY$_\alpha$

Finally, consider WJISP$_m$ with equal weights per job. If there was an interval i' in S belonging to the same job as i at the time when GREEDY$_\alpha$ processed i, charge $w(i)$ to i'. Otherwise, distribute the weight of i among the intervals in Q_i in the same way as above. It is easy to see that every interval $j \in T$ is charged at most $w(j)$ by intervals from the same job and at most $\frac{1}{\alpha}w(j)$ by other intervals. Therefore, we get $w(OPT) \leq (1+\frac{1}{\alpha})w(T) \leq \frac{1+\alpha}{\alpha(1-\alpha)}w(A)$.

The analysis of the approximation ratio of GREEDY$_\alpha$ is tight for all three cases. For example, consider the instance of WJISP$_1$ with equal weights per job shown in Fig. 2. Intervals in the same row belong to the same job, and the labels correspond to the weights of the intervals, where $b = \frac{1}{\alpha}$ and ε is a very small positive value. Every group of three intervals is called a *phase*; the shown example consists of 3 phases. In an example with k phases, GREEDY$_\alpha$ will output a single interval of weight b^{k-1} (shown dashed in Fig. 2), while the optimal solution contains $2k$ intervals (shown bold in Fig. 2) of total weight $\sum_{i=0}^{k-1}(b^i + b^{i+1} - \varepsilon) = (1+b)\frac{b^k-1}{b-1} - k\varepsilon$. Simple calculations show that the approximation ratio of GREEDY$_\alpha$ tends to $\frac{\alpha(1-\alpha)}{1+\alpha}$ as $\varepsilon \to 0$ and $k \to \infty$.

Worst-case instances for WJISP with arbitrary weights can be constructed in a similar way. Furthermore, these constructions can be adapted to WJISP$_m$ for $m > 1$ by including m copies of every interval (putting each copy in a different job). $\square$

By optimizing the choice of α, we obtain the following corollary of Theorem 1.

Corollary 1. *For WJISP$_m$ with equal weights per job, GREEDY$_\alpha$ achieves approximation ratio* $3 - 2\sqrt{2} \approx 0.172$ *for* $\alpha = \sqrt{2} - 1 \approx 0.414$*. For WJISP$_m$ with arbitrary weights, GREEDY$_\alpha$ achieves approximation ratio* $\frac{1}{8}$ *for* $\alpha = \frac{1}{2}$.

The running time of GREEDY$_\alpha$ depends on how efficient the cheapest conflict set C_i can be determined. We use dynamic programming to show that this can be done in $O(\min(n \log n, nm))$ time, yielding a time complexity of $O(n^2 \min(\log n, m))$ for GREEDY$_\alpha$.

First, we explain the dynamic programming approach for determining C_i in time $O(nm)$. We use the notation from the proof of Theorem 1. It suffices to show how C'_i is computed. See Fig. 3. For every interval $j \in Q_i$, consider a point p just before the right endpoint of j. If p is contained in m intervals of Q_i, at least one of these intervals must be in C'_i. Let $p_1, \ldots, p_r$ be all such points (i.e., points just before the right endpoint

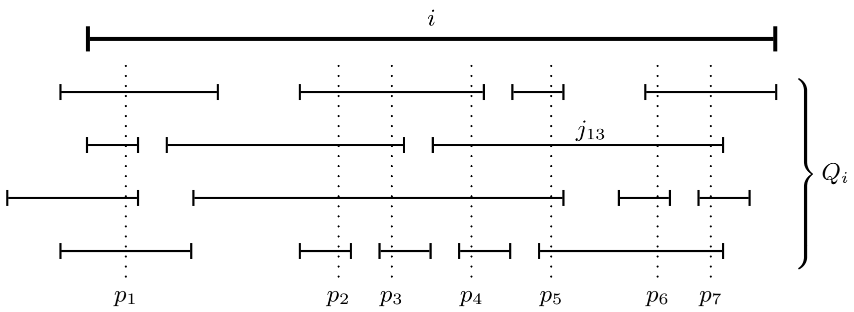

Fig. 3. Current interval i and intersecting intervals in S

$C(0) = 0$;
for $h = 1$ **to** r **do** $C(h) = \infty$; **od**;
for $\ell = 1$ **to** s **do**
    if $\mathit{first}(j_\ell)$ is not undefined **then**
        for $h = \mathit{first}(j_\ell)$ **to** $\mathit{last}(j_\ell)$ **do**
            $C(h) = \min\{C(h), C(\mathit{first}(j_\ell) - 1) + w(j_\ell)\}$;
        od;
    fi;
od;

Fig. 4. Dynamic programming algorithm

of an interval in Q_i) that are contained in m intervals of Q_i. C_i' is a minimum-weight subset of Q_i such that every point p_h, $1 \le h \le r$, is covered. (A set of intervals *covers* a point if the point is contained in at least one interval of that set.)

Let $j_1, j_2, \ldots, j_s$ denote the intervals in Q_i in order of non-decreasing right endpoints. For every interval $j_\ell \in Q_i$, let $\mathit{first}(j_\ell)$ and $\mathit{last}(j_\ell)$ be the smallest resp. largest index of a point p_h that is contained in j_ℓ. (If j_ℓ does not contain any point p_h, $\mathit{first}(j_\ell)$ and $\mathit{last}(j_\ell)$ are undefined.) In Fig. 3, we have $\mathit{first}(j_{13}) = 4$ and $\mathit{last}(j_{13}) = 7$, for example. The points $p_1, \ldots, p_r$, the intervals $j_1, \ldots, j_s$, and the tables *first* and *last* can all be constructed in time $O(n)$. Then we use the dynamic programming procedure shown in Fig. 4 to compute values $C(h)$, $1 \le h \le r$.

Claim. After ℓ iterations of the outer for-loop, it holds for $1 \le h \le r$ that $C(h)$ is the weight of a minimum-weight subset of $\{j_1, \ldots, j_\ell\}$ that covers $p_1, \ldots, p_h$ (or ∞ if no such subset exists).

This claim can be proved easily by induction. It implies that $C(r) = w(C_i')$ at the end of the execution of the procedure, and additional bookkeeping will allow to construct C_i' at no extra cost. The running-time of the procedure in Fig. 4 is $O(rm) = O(nm)$, because the body of the inner for-loop is executed exactly rm times. (For each of the r points p_h, the body of the inner for-loop is executed only for the m intervals containing p_h.)

In order to get an implementation that runs in $O(n \log n)$ time, we observe that the costs $C(h)$, $0 \leq h \leq r$, are always non-decreasing. This implies that one does not have to enumerate over $h = \mathit{first}(j_\ell)$ up to $h = \mathit{last}(j_\ell)$, but instead can find the "breakpoint value" for h (say h^*) such that $C(h)$ remains unchanged for $h < h^*$ and is updated to $C(\mathit{first}(j_\ell) - 1) + w(j_\ell)$ for all $h^* \leq h \leq \mathit{last}(j_\ell)$. We use a balanced search tree T that stores elements of the form $(h, C(h))$, sorted by h (and thus also by $C(h)$) in non-decreasing order. If T contains $(h_1, C(h_1))$ and $(h_2, C(h_2))$ but no pair $(h', C(h'))$ with $h_1 < h' < h_2$, this means that $C(h) = C(h_1)$ for $h_1 \leq h < h_2$. In each iteration, we use T to determine h^* in time $O(\log n)$, insert $(h^*, C(h^*))$ into T, and remove invalid pairs $(h, C(h))$ from T. See [8] for details.

Q_i constitutes an m-colorable interval graph, and $Q_i \setminus C'_i$ is a maximum-weight $(m-1)$-colorable subgraph of Q_i. So we have the following corollary:

Corollary 2. *A maximum weight k-colorable subgraph in a $k+1$ colorable interval graph can be obtained in $O(n \min(\log n, k))$ time.*

This is a speedup compared to the $O(kS(n))$ algorithm for the general case of the maximum weight k-colorable subgraph problem in interval graphs ([6]) (where $S(n)$ is the running time for finding a shortest path in graphs with $O(n)$ edges).

3 Competitive lower bounds

We are interested in the best approximation ratio that can be achieved by any myopic algorithm for WJISP, and we use competitive analysis to answer this question. We phrase our arguments in terms of an adversary who constructs worst-case instances for a myopic algorithm incrementally, depending on previous decisions made by the algorithm. Thus, we present the construction of worst-case examples for myopic algorithms as a game played by the adversary and the algorithm.

Theorem 2. *No myopic algorithm for JISP$_m$ can achieve an approximation ratio better than $\frac{1}{2}$. No myopic algorithm for WJISP$_1$ with equal weights per job can achieve an approximation ratio better than $3 - 2\sqrt{2} \approx 1/5.828 \approx 0.172$. No myopic algorithm for WJISP$_1$ with arbitrary weights can achieve an approximation ratio better than $1/7.103 \approx 0.141$.*

Proof. We restrict ourselves here to proving only the first two claims of the theorem. The proof of the third claim uses a similar approach as the proof of the second claim, but is more complicated. Details can be found in [8].

Consider the following construction for JISP$_m$. Initially, the adversary presents a set Q of $2m$ intervals with the same right endpoint p. Let S be the set of at most m intervals that are selected by the algorithm at this time. Let S' be a subset of Q that contains all intervals in S and that has cardinality m. For every interval in S', the adversary presents an interval that belongs to the same job and that is to the right of p. The optimal solution contains $2m$ intervals, while the algorithm accepts at most m intervals. This proves the first claim of the theorem. All intervals in this construction can have equal length.

Now consider WJISP$_1$ with equal weights per job. When the adversary presents an interval i to the algorithm, we say that the adversary *plays* i. We can assume that the adversary knows exactly for which values of $w(i)$ the algorithm would accept i and for which values of $w(i)$ the algorithm would reject i. Assume that the algorithm has currently selected a single interval c and that the adversary is about to present an interval i that intersects c. Let $A(i)$ be the set of all possible weights $w(i)$ for which the algorithm would accept i, and let $R(i)$ be the set of all possible weights $w(i)$ for which the algorithm would reject i. We can assume without loss of generality that $A(i) \neq \emptyset$, $R(i) \neq \emptyset$, and that $\sup R(i)$ is finite; otherwise, the approximation ratio of the algorithm would be unbounded. We would like to define $la(i)$ ("lightest accept") as the smallest value of $w(i)$ for which the algorithm would accept i and $hr(i)$ ("heaviest reject") as the largest value of $w(i)$ for which the algorithm would reject i. More formally, these values are defined as follows, where $\delta_i > 0$ is a parameter that depends on i. If $\min A(i)$ exists, let $la(i) = \min A(i)$. Otherwise, let $la(i)$ be an arbitrary value in $A(i)$ such that $|la(i) - \inf A(i)| < \delta_i$. If $\max R(i)$ exists, let $hr(i) = \max R(i)$. Otherwise, let $hr(i)$ be an arbitrary value in $R(i)$ such that $|hr(i) - \sup R(i)| < \delta_i$. As $\sup R(i) \geq \inf A(i)$, we get

$$la(i) \leq hr(i) + 2\delta_i. \tag{3}$$

At any time of the game, the following conditions will hold: exactly one interval c of weight $w(c)$ is currently selected by the algorithm; among the intervals that were presented to the algorithm before c, there is a subset $P \neq \emptyset$ of independent intervals with total weight $w(P) > 0$ such that no interval in P belongs to the same job as c; the rightmost right endpoint of an interval in P is to the left of the right endpoint of c. We use $x := w(P)/w(c)$ as an indicator for the *progress* made by the adversary so far; note that the ratio of the weight of an optimal solution to the weight of the solution of the algorithm is at least x at all times (equivalently, the approximation ratio of the algorithm is no better than $1/x$).

Let $\varepsilon > 0$ be a small positive constant. We require $\varepsilon < 1/37$. The adversary tries to show that the approximation ratio of the algorithm is no better than $(1+40\varepsilon)/(3+2\sqrt{2})$. As we can choose ε arbitrarily small and as $1/(3 + 2\sqrt{2}) = 3 - 2\sqrt{2}$, this implies the second claim of the theorem. The main part of the game consists of a number of steps. We will ensure that in every step either x increases by the factor $1/(1 - \varepsilon)$ (we say that *the adversary has made progress* when this happens) or the game ends in favor of the adversary (i.e., the ratio of the algorithm is $(1 + 40\varepsilon)/(3 + 2\sqrt{2})$ or worse). The strategy used by the adversary in every step is sketched in Fig. 5. Here, b is implicitly understood to be an interval belonging to the same job as c and to the right of all previously presented intervals. At the beginning of the step, a is understood to be an interval that belongs to a new job and that intersects c, but no interval presented before c. Finally, a' is an interval that belongs to a new job and that intersects a, b and c, but no interval presented before c. See Fig. 6 for a sketch of a, b, c and a'.

Initially, the adversary presents an interval i_0 of weight $w(i_0) = 1$. The algorithm must accept this interval. Then the adversary presents an interval i_1 of weight $w(i_1) = 6$ that belongs to a new job, that intersects i_0, and whose right endpoint is to the right of the right endpoint of i_0. The algorithm must accept i_1 and preempt i_0; otherwise, its approximation ratio would be no better than $1/6$. We let $c = i_1$ and $P = \{i_0\}$, thus getting $x = w(P)/w(c) = 1/6$. This completes the initialization of the game; the remainder of the game consists of steps that are carried out as sketched in Fig. 5.

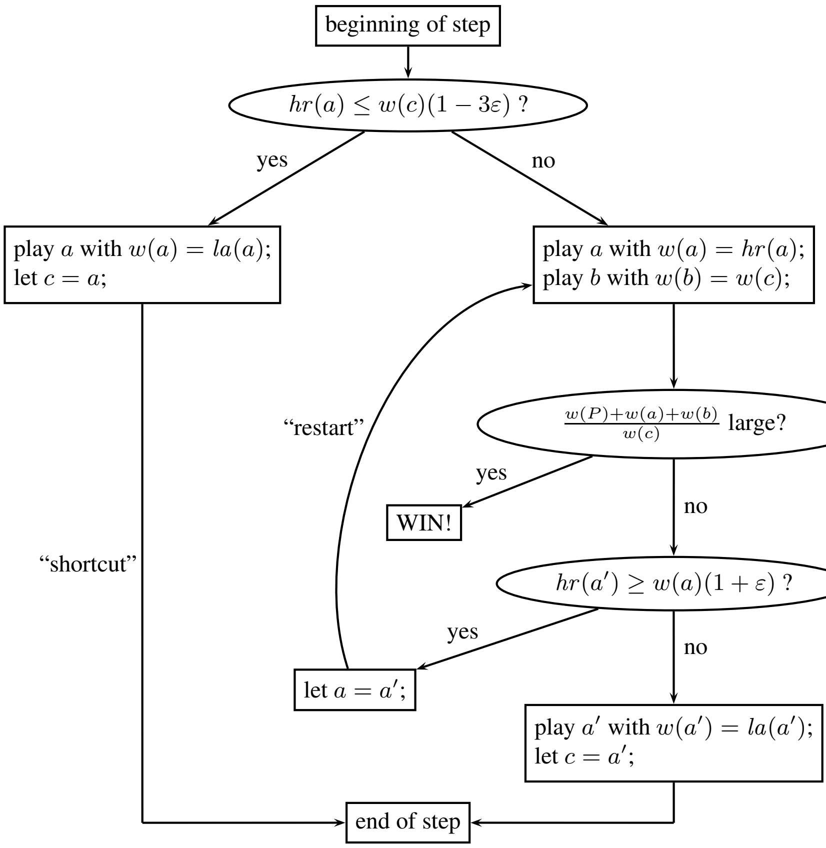

Fig. 5. Adversary strategy for WJISP_1 with equal weights per job

We explain one step of the game in detail. Let a be an interval of a new job that intersects c, but no earlier interval. For the definition of $hr(a)$ and $la(a)$, let $\delta_a = \varepsilon w(c)$. If $hr(a) \le w(c) - 3\delta_a = w(c)(1-3\varepsilon)$, set $w(a) = la(a)$. Note that $la(a) \le w(c) - \delta_a$ by (3). The adversary plays a; the algorithm accepts a and preempts c. Now a becomes the current interval, and the adversary has made progress because $w(P)/w(a) \ge w(P)/(w(c) - \delta_a) = w(P)/w(c) \cdot 1/(1-\varepsilon)$. The adversary proceeds to the next step.

Now assume that $hr(a) > w(c) - 3\delta_a$. The adversary sets $w(a) = hr(a)$ and presents a to the algorithm, and the algorithm rejects a. Let $\alpha = w(a)/w(c)$, and note that $\alpha \ge 1 - 3\varepsilon$. Next, the adversary plays an interval b that is to the right of all previous intervals and that belongs to the same job as c. As we have equal weights per job, we must have $w(b) = w(c)$. The algorithm can either accept b (and preempt c) or not; this does not affect the adversary's strategy.

At this time, an optimal solution has weight at least $w(P)+w(a)+w(b) = w(P)+(1+\alpha)w(c)$, while the algorithm has accepted only the interval (b or c) with weight $w(c)$. Therefore, the ratio of the weight of an optimal solution to the weight of the

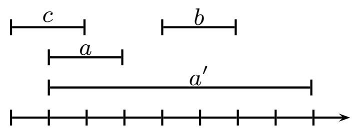

Fig. 6. Intervals played in a step without restart and shortcut. At the beginning of the step, only c is accepted by the algorithm; during the step, the algorithm rejects a, accepts or rejects b, and accepts a'; at the end of the step, only a' is accepted by the algorithm

solution produced by the algorithm is at least $x + 1 + \alpha$ at this time. Let us denote this ratio by β, i.e., $\beta = \frac{w(P)+w(a)+w(b)}{w(c)} \geq x + 1 + \alpha$.

We say that β is large if $\beta \geq (3+2\sqrt{2})/(1+40\varepsilon)$. If β is large, the adversary stops and wins the game: the approximation ratio of the algorithm is no better than $1/\beta$. Assume that β is not large. Then the adversary plays an interval a' that belongs to a new job and that intersects a, b and c, but no interval played before c. For the definition of $hr(a')$ and $la(a')$, let $\delta_{a'} = \varepsilon w(a)$. If $hr(a') \geq w(a) + \delta_{a'} = w(a)(1+\varepsilon)$, the adversary sets $w(a') = hr(a')$, plays a', and restarts the current step with a' in place of a (and with b in place of c, if the algorithm has accepted b). Here, restarting the current step means continuing as if a' instead of a had been played as the first interval of the current step. Note that $w(a') \geq w(a)(1+\varepsilon)$; therefore, there can be only finitely many restarts before the approximation ratio of the algorithm (which is smaller than $w(c)/w(a)$) becomes worse than, say, $1/6$.

If $hr(a') < w(a) + \delta_{a'}$, the adversary sets $w(a') = la(a')$ and plays a'. Note that $la(a') \leq w(a) + 3\delta_{a'}$ by (3). The algorithm accepts a', and a' becomes the new current interval. Observe that $P \cup \{a, b\}$ is a set of independent intervals. The progress of the adversary is now

$$\frac{w(P)+w(a)+w(b)}{w(a')} \geq \frac{w(P)+w(a)+w(b)}{w(a)+3\delta_{a'}} = \frac{\frac{x}{\alpha}+1+\frac{1}{\alpha}}{1+3\varepsilon}.$$

If $\frac{\frac{x}{\alpha}+1+\frac{1}{\alpha}}{1+3\varepsilon} \geq \frac{x}{1-\varepsilon}$, the adversary has made progress, and the next step begins with current interval a'. Assume that the adversary has not made progress. Then we must have $\frac{\frac{x}{\alpha}+1+\frac{1}{\alpha}}{1+3\varepsilon} < \frac{x}{1-\varepsilon}$. This implies

$$\frac{1}{\alpha} < \frac{x\frac{1+3\varepsilon}{1-\varepsilon} - 1}{x+1}. \tag{4}$$

If $x\frac{1+3\varepsilon}{1-\varepsilon} - 1 \leq 0$, this cannot happen. Therefore, we must have

$$x > \frac{1-\varepsilon}{1+3\varepsilon}. \tag{5}$$

We obtain from (4) that $\alpha > (x+1)/(x\frac{1+3\varepsilon}{1-\varepsilon} - 1)$. Therefore,

$$\beta \geq x + 1 + \alpha \geq x + 1 + \frac{x+1}{x\frac{1+3\varepsilon}{1-\varepsilon} - 1} = \frac{x^2+x}{x - \frac{1-\varepsilon}{1+3\varepsilon}}.$$

We claim that $(x^2+x)/(x-\frac{1-\varepsilon}{1+3\varepsilon})$ is large, implying that the adversary has actually won the game already before presenting a'.

First, consider the case that $x \leq 1.1$. As $\varepsilon \leq 1/37$, we have $\frac{1-\varepsilon}{1+3\varepsilon} = 1 - \frac{4}{1/\varepsilon+3} \geq 0.9$. Therefore, we get $x - \frac{1-\varepsilon}{1+3\varepsilon} \leq 1.1 - 0.9 = 0.2$ and, by (5), $x^2 + x \geq 0.9^2 + 0.9 = 1.71$. This implies that $\beta \geq 1.71/0.2 = 8.55$. Obviously, β is large.

Now consider the case that $x > 1.1$. We get: $x - \frac{1-\varepsilon}{1+3\varepsilon} \leq x - 1 + 4\varepsilon = (x-1)(1+40\varepsilon) - 40\varepsilon x + 44\varepsilon \leq (x-1)(1+40\varepsilon)$. Therefore, we obtain

$$\frac{x^2+x}{x-\frac{1-\varepsilon}{1+3\varepsilon}} \geq \frac{x^2+x}{x-1} \cdot \frac{1}{1+40\varepsilon}.$$

Is is easy to show that the function $f(x) = \frac{x^2+x}{x-1}$, where x is in the range $[1.1, \infty)$, is minimized for $x = 1+\sqrt{2}$ with $f(1+\sqrt{2}) = 3+2\sqrt{2}$. Therefore, β is large, and the adversary won the game already before a' was played. $\square$

References

1. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation. Combinatorial Optimization Problems and their Approximability Properties*. Springer, Berlin, 1999.
2. A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. S. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing STOC'00*, pages 735–744, 2000.
3. A. Bar-Noy, S. Guha, J. S. Naor, and B. Schieber. Approximating the throughput of multiple machines under real-time scheduling. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing STOC'99*, pages 622–631, 1999.
4. P. Berman and B. DasGupta. Improvements in throughput maximization for real-time scheduling. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing STOC'00*, pages 680–687, 2000.
5. P. Berman, Z. Zhang, J. Bouck, and W. Miller. Aligning two fragmented sequences. Manuscript, 1999.
6. M. C. Carlisle and E. L. Lloyd. On the k-coloring of intervals. *Discrete Appl. Math.*, 59:225–235, 1995.
7. Y. Crama, O. Flippo, J. van de Klundert, and F. Spieksma. The assembly of printed circuit boards: a case with multiple machines and multiple board types. *European Journal of Operational Research*, 98:457–472, 1997.
8. T. Erlebach and F. Spieksma. Simple algorithms for a weighted interval scheduling problem. Technical Report M00-01, Department of Mathematics, Maastricht University, April 2000. http://www.tik.ee.ethz.ch/∼erlebach/Report-M00-01.ps.gz.
9. S. A. Goldman, J. Parwatikar, and S. Suri. Online scheduling with hard deadlines. *Journal of Algorithms*, 34(2):370–389, 2000.
10. M. H. Goldwasser. Patience is a virtue: The effect of slack on competitiveness for admission control. In *Proceedings of the 10th Annual ACM–SIAM Symposium on Discrete Algorithms SODA'99*, pages 396–405, 1999.
11. D. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Company, Boston, 1997.
12. F. Spieksma. On the approximability of an interval scheduling problem. *Journal of Scheduling*, 2:215–227, 1999.